AMENDMENTS TO THE CLAIMS

1. (Currently amended) A method of dynamically converting binary executable program code of a subject computing architecture into binary target code executed on a target computing system, wherein conversion of the program code is interleaved with execution of the target code on the target computing system, comprising the steps of:

decoding a plurality of instructions in the program code;

determining which type of nodes to generate in an intermediate representation for each of the decoded instructions in the program code, including determining that one or more of the decoded instructions require base nodes having basic RISC-like functionality which provides an expanded representation of semantics of the decoded instructions, and one or more of the decoded instructions require complex nodes having complex CISC-like functionality which provides a compact representation of semantics of the decoded instructions;

generating the intermediate representation of the decoded instructions using the determined types of nodes, including generating the base nodes and the complex nodes in the intermediate representation from the respective decoded instructions;

generating the target code from the intermediate representation; and

executing the target code on the target computing system.

2. (Cancelled)

3. (Cancelled)

4. (Previously presented) The method of claim 1, wherein the base nodes are generic across a plurality of possible subject computing architectures.

5. (Cancelled)

6. (Previously presented) The method of claim 1, wherein the determining step includes that the program code includes immediate type instructions in which a constant operand value is encoded into the instruction in an immediate field and in response determining that the immediate type instructions require the complex nodes.

7. (Previously presented) The method of claim 1, wherein each of the complex nodes may be decomposed into a plurality of the base nodes to represent the same semantics of an instruction in the decoded program code.

8. (Previously presented) The method of claim 1, further comprising the step of generating the complex nodes only for those features which are correspondingly configurable on the subject computing architecture.

9. (Cancelled)

10. (Cancelled)

11. (Cancelled)

12. (Cancelled)

13. (Cancelled)

14. (Currently amended) The method of claim 1, wherein ~~the plurality of possible types of nodes further include~~ the determining step further comprises determining that one or more of the decoded instructions require architecture specific nodes.

15. (Currently amended) The method of claim 14, wherein the ~~program code is dynamically translated into the~~ target code relates to ~~for execution on the target computing system having~~ a target architecture, said method further comprising:

generating the intermediate representation to include the architecture specific nodes which are specific to a particular combination of the subject computing architecture and the target architecture.

16. (Currently amended) The method of claim 15, wherein the ~~generating step~~ method further comprises:

initially representing all of the instructions in the program code as subject architecture specific nodes, where each subject architecture specific node corresponds to a respective instruction in the program code;

determining whether an instruction in the program code is one in which to provide a target architecture specialized conversion function;

converting the subject architecture specific nodes into target architecture specific nodes for those instructions determined to provide ~~a~~ the target architecture specialized conversion function; and

generating the base nodes from the remaining subject architecture specific nodes which are not identified as providing the target architecture specialized code generation function.

17. (Previously presented) The method of claim 16, further comprising generating the target code from the target architecture specific nodes, wherein the target code is specialized for the target architecture.

18. (Previously presented) The method of claim 15, further comprising generating the target code from the base nodes, wherein the target code is not specialized for the target architecture.

19. (Cancelled)

20. (Currently amended) A computer readable storage medium having translator software resident thereon in the form of computer readable code executable by a target computer system to ~~perform~~ dynamically convert binary executable program code of a subject computing architecture into binary target code executed on the target computing system, wherein conversion of the program code is interleaved with execution of the target code on the target computing system, by performing the steps of:

decoding a plurality of instructions in a program code of a subject computing architecture;

determining which type of nodes to generate in an intermediate representation for each of the decoded instructions in the program code, including determining that one or more of the decoded instructions require base nodes having basic RISC-like functionality which provides an expanded representation of semantics of the decoded instructions, and one or more of the decoded

instructions require complex nodes having complex CISC-like functionality which provides a compact representation of semantics of the decoded instructions;

generating the intermediate representation of the decoded instructions using the determined types of nodes, including generating the base nodes and the complex nodes in the intermediate representation from the respective decoded instructions;

generating the target code from the intermediate representation; and

executing the target code on the target computing system.

21. (Cancelled)

22. (Cancelled)

23. (Previously presented) The computer readable storage medium of claim 20, wherein the base nodes are generic across a plurality of possible subject computing architectures.

24. (Cancelled)

25. (Currently amended) The computer readable storage medium of claim 20, wherein the determining step includes determining that the program code includes immediate type instructions in which a constant operand value is encoded into the instruction in an immediate field and in response determining that the immediate type instructions require the complex nodes.

26. (Previously presented) The computer readable storage medium of claim 20, wherein each of the complex nodes may be decomposed into a plurality of the base nodes to represent the same semantics of an instruction in the decoded program code.

27. (Previously amended) The computer readable storage medium of claim 20, further comprising the step of generating the complex nodes only for those features which are correspondingly configurable on the subject computing architecture.

28. (Cancelled)

29. (Cancelled)

30. (Cancelled)

31. (Cancelled)

32. (Cancelled)

33. (Currently amended) The computer readable storage medium of claim 20, wherein ~~the plurality of possible types of nodes further include~~ the determining step further comprises determining that one or more of the decoded instructions require architecture specific nodes.

34. (Currently amended) The computer readable storage medium of claim 33, wherein the ~~program code is dynamically translated into the~~ target code relates to ~~for execution on the target computing system having~~ a target architecture, said translator software further containing computer readable code executable by a computer to perform the following steps:

generating the intermediate representation to include the architecture specific nodes which are specific to a particular combination of the subject computing architecture and the target architecture.

35. (Currently amended) The computer readable storage medium of claim 34, said translator software further containing computer readable code executable by a computer to perform the following steps:

initially representing all of the instructions in the program code as subject architecture-specific nodes, where each subject architecture specific node corresponds to a respective instruction in the program code;

determining whether an instruction in the program code is one in which to provide a target architecture specialized conversion function;

converting the subject architecture specific nodes into target architecture specific nodes for those instructions determined to provide ~~a~~ the target architecture specialized conversion function; and

generating the base nodes from the remaining subject architecture specific nodes which are not identified as providing the target architecture specialized code generation 14 function.

36. (Previously presented) The computer readable storage medium of claim 35, wherein said translator software contains computer readable code executable by a computer to generate the target code from the target architecture specific nodes which is specialized for the target architecture.

37. (Previously presented) The computer readable storage medium of claim 34, said translator software further containing computer readable code executable by a computer to generate the target code from the base nodes which is not specialized for the target architecture.

38. (Currently amended) A translator apparatus ~~for use in a target computing environment~~ having a processor and a memory coupled to the processor ~~for converting~~ to form a target computing environment, the translator apparatus being arranged to dynamically convert binary executable subject program code appropriate to a subject computing architecture to produce binary target code appropriate to the target computing environment, wherein conversion of the program code is interleaved with execution of the target code in the target computing environment, the translator apparatus comprising:

a decoding mechanism to decode a plurality of instructions in the subject program code;

a type determining mechanism to determine which type of nodes to generate in an intermediate representation for each of the decoded instructions in the program code, including determining that one or more of the decoded instructions require base nodes having basic RISC-like functionality which provides an expanded representation of semantics of the decoded instructions, and one or more of the decoded instructions require complex nodes having complex CISC-like functionality which provides a compact representation of semantics of the decoded instructions;

an intermediate representation generating mechanism to generate the intermediate representation of the decoded instructions using the determined types of nodes, including generating the base nodes and the complex nodes in the intermediate representation from the respective decoded instructions;

a target code generating mechanism to generate the target code from the intermediate representation; and

a target code execution mechanism to execute the target code on the processor in ~~of~~ the target computing environment.

39. (Cancelled)

40. (Cancelled)

41. (Currently amended) The translator apparatus of claim 38, wherein the base nodes are generic across a plurality of possible subject computing architectures.

42. (Cancelled)

43. (Previously presented) The translator apparatus of claim 38, wherein the complex nodes represent immediate type instructions in which a constant operand value is encoded into the immediate type instruction itself in an immediate field.

44. (Previously presented) The translator apparatus of claim 38, wherein the complex nodes may be decomposed into a plurality of the base nodes to represent the same semantics of an instruction in the decoded program code.

45. (Previously presented) The translator apparatus of claim 38, wherein the program code is designed to be executed by a subject architecture, the intermediate representation generating mechanism further comprises a complex node generating mechanism for generating the complex nodes only for those features correspondingly configurable on the subject computing architecture.

46. (Cancelled)

47. (Cancelled)

48. (Cancelled)

49. (Cancelled)

50. (Cancelled)

51. (Currently amended) The translator apparatus of claim 38, wherein ~~the plurality of possible types of nodes further include~~ the type determining mechanism is further arranged to determine that one or more of the decoded instructions require architecture specific nodes.

52. (Currently amended) The translator apparatus of claim 51, said intermediate representation generating mechanism further comprising:

an architecture specific node generating mechanism for generating the intermediate representation to include the architecture specific nodes which are specific to a particular combination of ~~a~~ the subject computing architecture and ~~a~~ the target architecture.

53. (Currently amended) The translator apparatus of claim 52, the intermediate representation generating mechanism being configured to:

initially represent all of the instructions in the subject program code as subject architecture-specific nodes, where each subject architecture specific node corresponds to a respective instruction in the subject program code;

determine whether an instruction in the subject program code is one in which to provide a target architecture specialized conversion function;

convert the subject architecture specific nodes into target architecture specific nodes for those instructions determined to provide ~~a~~ the target architecture specialized conversion function; and

generate the base nodes from the remaining subject architecture specific nodes which are not identified as providing a target architecture specialized code generation function.

54. (Previously presented) The translator apparatus of claim 52, further comprising a specialized target code generating mechanism for generating the target code from the target architecture specific nodes which is specialized for the target architecture.

55. (Previously presented) The translator apparatus of claim 52, further comprising a non specialized target code generating mechanism for generating the target code from the base nodes which is not specialized for the target architecture.

56.-85. (Cancelled)